Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 9, n. 1, 2022.

Trabalho apresentado no XLI CNMAC, Unicamp - Campinas - SP, 2022.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics
Preprint

Sobre a aplicação de homomorfismo na criptografia

Aline de L. Z. Lunkes[1], Fábio Borges[2]
LNCC, Petrópolis, RJ

Resumo. O uso de novas tecnologias e da internet vem crescendo, por isso, devemos garantir a privacidade e a segurança dos usuários ao navegar na web. Com este intuito, o esquema de criptografia depende do compartilhamento de uma chave entre os pares envolvidos na troca de uma mensagem. O conceito de homomorfismo foi utilizado como uma possível solução para a computação sem a necessidade de decifrar os dados. A Criptografia Homomórfica é o esquema de criptografia que preserva a privacidade e segurança dos dados criptografados.

Palavras-chave. Criptografia, Homomorfismo, Computação, Privacidade, Segurança

1 Introdução

A criptografia utiliza técnicas de computação e matemática para transformar informações em códigos, ou seja, técnicas para comunicação segura na presença de terceiros, chamados atacantes ou Eve, um exemplo de como isso ocorre está na figura 1, ou a computação em nuvem, que executa determinadas funções computáveis nos dados, preservando os recursos da função utilizada e o formato dos dados criptografados. Além disso, é preciso de uma maneira para computar tais dados, por exemplo, com operações de adição e multiplicação. Em alguns esquemas são executados apenas operações de adição ou multiplicação, por exemplo, no algoritmo de Paillier e de RSA, respectivamente.

Com o crescimento no uso de tecnologias, tanto na indústria quanto no cotidiano, e a demanda de aplicativos para facilitar a conexão nas redes sociais, transações bancárias, armazenamentos de arquivos, seja em celulares ou computadores, salientamos uma preocupação com a segurança e privacidade dos dados. Deste modo, apresentamos neste trabalho a introdução de Criptografia Homomórfica, no inglês Homomorphic Encryption - HE. Para o uso de HE, necessitamos entender o conceito e as aplicações do Homomorfismo.

Além disso, um esquema de HE é constituído por quatro algoritmos, a saber, 1) Geração das chaves pública e secreta; 2) Cifrar $E(m)$; 3) Decifrar $D(E(m))$ e 4) Homomorfismo. No processo de geração das chaves retorna ou um par de chaves secretas e públicas para a versão assimétrica (conhecida como Criptografia de Chave Pública), ou uma única chave para a versão simétrica. O algoritmo de Homomorfismo é uma operação específica de HE e foi introduzido pelos autores [6]. Tendo como entrada os textos cifrados das mensagens m_1 e m_2, $E(m_1)$ e $E(m_2)$, e como saída o texto cifrado de uma função f sobre m_1 e m_2, $E(f(m_1, m_2))$. Dessa maneira, a operação f é calculada sobre as mensagens sem ser necessário decifrar antecipadamente.

Um ponto importante na HE é que o formato das mensagens cifradas, após o processo de Homomorfismo, deve ser preservado para que as mensagens sejam decifradas corretamente, ou seja, a definição de Homomorfismo nos garante que isso ocorra. Além disso, é permitido um crescimento logarítmico do erro no texto cifrado com o número de operações Homomórficas realizadas, ver [2], ou muitas vezes chamado de profundidade do circuito.

[1] alunkes@lncc.br
[2] borges@lncc.br

Além disso, o termo Cliente-Servidor é comum utilizado em computação, neste trabalho denotamos por Alice e Bob, respectivamente, ver [8]. O Cliente é quem deseja enviar mensagem ou dados para o servidor, mas não compartilha nenhum de seus recursos com o mesmo. O Servidor é quem processa os dados ou armazenas e quando solicitado pelo Cliente, executa algumas funções e retorna para o Cliente. Na HE o Cliente envia os dados cifrados para o servidor, e o mesmo não decifra os dados, apenas computa o que o Cliente deseja.

Um exemplo de Criptografia de Chave Pública é a assinatura digital, podemos utilizar o algoritmo homomórfico multiplicativo de RSA, Alice (Cliente) deseja assinar um documento enviado por Bob (Servidor), utilizando a chave pública de Alice, ver [7]. Pois, é a portadora da chave privada para assinar, e a assinatura cifrada de Alice é decifra com a chave pública, garantindo a sua autenticidade na assinatura. Temos algoritmos de HE que são Homomorficos aditivos e são utilizados em votação eletrônica, por exemplo os algoritmos de Paillier, ver [5] e o de Damgård-Jurik[3]. Além disso, temos o esquema de Criptografia Totalmente Homomórfica, no inglês Fully Homomorphic Encryption - FHE, que permite executar as operações homomórficas aditivas e multiplicativas em um único algoritmo, por exemplo o algoritmo de NTRU, ver [4] .

2 Homomorfismo

Nesta seção tratamos do conceito de Homomorfismo, que será utilizado ao longo deste trabalho. Um Homomorfismo entre duas estruturas algébricas de mesmo tipo é uma aplicação que preserva essas estruturas. Por exemplo, as estruturas podem ser grupos, anéis ou espaços vetoriais, com suas respectivas operações.

Por exemplo, se G e H têm a adição, $+$, para ambas as estruturas, então um Homomorfismo f entre G e H satisfaz

$$f(x+y) = f(x) + f(y)$$

para quaisquer x, $y \in G$.

Agora, se G e H são ambos anéis, com uma operação extra de multiplicação, $\cdot$, então, além da igualdade anterior, o Homomorfismo f ainda satisfaz

$$f(x \cdot y) = f(x) \cdot f(y)$$

para quaisquer x, $y \in G$.

Vamos apresentar dois exemplos de Homomorfismo, o primeiro está associado a anéis, e o segundo a grupos.

Exemplo 2.1. *Sejam G, H dois anéis quaisquer com operações $+, \cdot$. Seja a aplicação $f : G \to H$ dada por $f(x) = 0_H$ para qualquer $x \in G$, onde 0_H é o elemento neutro da adição em H. Então, claramente essa aplicação é um homomorfismo.*

Uma das aplicações de homomorfismo é a sua utilização para garantir que dois anéis sejam isomorfos entre si, ou seja, tenham operações equivalentes e o homomorfismo entre eles é ainda uma bijeção. Assim, se $f : A \to B$ é um homomorfismo entre anéis, temos que a imagem de f como um subanel de B é isomorfa ao anel quociente $A/\ker(f)$, onde $\ker(f)$ é o kernel, ou núcleo, de f. Mais informações sobre esse resultado e suas consequências podem ser obtidas em [3] e [1].

Exemplo 2.2. *Seja G um grupo não vazio com operação $\cdot$. Seja $g \in G$ um elemento fixado de G, e definimos a aplicação $\varphi : G \to G$ dada por $\varphi(x) = g \cdot x \cdot g^{-1}$, para qualquer $x \in G$.*

[3] `http://security.hsr.ch/msevote/damgardjurik`

Sejam $x, y \in G$ dois elementos quaisquer, então, pela associatividade e pelo elemento neutro do grupo, temos

$$\varphi(x) \cdot \varphi(y) = (g \cdot x \cdot g^{-1}) \cdot (g \cdot y \cdot g^{-1}) = g \cdot (x \cdot y) \cdot g^{-1} = \varphi(x \cdot y).$$

Logo, φ é um Homomorfismo.

3 Criptografia Homomórfica

Nesta seção tratamos do esquema de Criptografia Homomórfica, em inglês Homomorphic Encryption, denotado por HE, que é composto por quatro algoritmos, a saber,

a) **Geração de chaves,** em inglês *KeyGen*, onde geramos as chaves pública e privada. Estas chaves são utilizadas para cifrar e decifrar as mensagens, ou os dados entre o cliente e o servidor. A chave pública é de conhecimento de todos, e a chave privada é de conhecimento apenas do proprietário.

   Na geração de chaves para a assinatura digital são executadas duas funções, autenticação e a encriptação. Na autenticação de mensagens utiliza a função matemática criptográfica de *hash*, suas entradas são de tamanhos arbitrários, podendo ser até em *GigaBytes* e após cifrarmos suas saída apresentam valores padronizados, e de tamanho fixo. Tornando difícil encontrar um valor de *hash* para ter acesso a mensagem, produzindo assim, um resumo.

   Para a encriptação utiliza o resumo, a saída do *hash*, e a chave privada resultando na assinatura digital. Assim, para verificar a sua autenticidade na assinatura é utilizado um software que faz as seguintes verificações: 1) calcula a função de *hash* da mensagem; 2) decifra a assinatura com a chave pública do assinante; e 3) compara o resultado computado com o decifrado.

b) **Cifrar,** em inglês *Encryption*, denotamos por $E(m)$, onde m é a mensagem a ser cifrada, ou o dado a ser escondido. Neste algoritmo utilizamos a chave pública.

c) **Decifrar,** em inglês *Decryption*, denotado por $D(E(m))$, onde $E(m)$ é a mensagem a ser decifrada com o auxílio da chave privada.

d) **Homomorfismo,** em inglês *Eval*, é uma aplicação específica do HE, que leva mensagens cifradas em mensagens cifradas, preservando as suas operações executadas.

Em um exemplo de criptografia, temos o Cliente (Alice) que deseja enviar uma mensagem para o Servidor (Bob). Necessitando assim de chaves públicas e privadas para ocorrer esta troca de mensagens. Deste modo, é gerado um par de chaves públicas (tais chaves são de fácil acesso a qualquer usuário) e privadas (estas chaves são acessadas somente por um usuário, o proprietário) para ambos. Alice faz o uso de HE para cifrar a mensagem, com a chave pública, e envia a mensagem para Bob. Por outro lado, Bob recebe a mensagem, e a decifra utilizando a sua chave privada. Porém, se existe um espião (Eve), entre o cliente e o servidor, em um canal inseguro, ele não consegue decodificar a mensagem, pois não tem acesso à chave privada.

Na HE, considerando o exemplo acima, para esta troca de mensagem segura, fazemos o uso do Homomorfismo. Ou seja, após Alice enviar a mensagem cifrada, é aplicado um Homomorfismo f sobre a mensagen cifrada ($E(m)$) sem o conhecimento de m, e obtém-se $f(E(m))$. Deste modo, a mensagem a ser decifrada corretamente é preservada. Na figura 1, mostramos como isso ocorre.

DOI: 10.5540/03.2022.009.01.0306

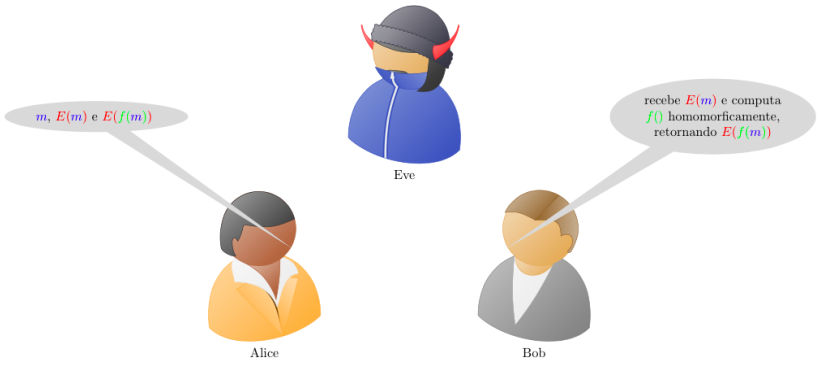

Figura 1: Homomorfismo no esquema de HE.

Na próxima subseção, apresentamos um algoritmo bem conhecido e que faz uso de HE, a saber RSA. Apresentamos primeiro alguns conceitos matemáticos que são importantes para o uso de HE.

Seja n um inteiro com $n > 1$. Então o anel modular $\mathbb{Z}_n$ herda algumas propriedades de $\mathbb{Z}$, pois é um anel quociente de $\mathbb{Z}$. Assim, considere o conjunto U_n formado pelas unidades de $\mathbb{Z}_n$. Ou seja, o conjunto $U_n = \{x \in \mathbb{Z}_n | x \text{ tem um inverso multiplicativo em } \mathbb{Z}_n\}$. Note que U_n também pode ser expresso como $U_n = \{x \in \mathbb{Z}_n | (x, n) = 1\}$, e que U_n forma um grupo multiplicativo. A ordem de U_n é denotada em geral por $\varphi(n)$. A função φ é chamada de função de *Euler*.

Os próximos resultados apresentam propriedades da função de *Euler*, e suas demonstrações podem ser obtidas em [3]. O próximo resultado é chamado de Pequeno Teorema de Fermat.

Teorema 3.1. *Seja p um primo e suponha que $x \in \mathbb{Z}$ satisfaça $(x, p) = 1$. Então $x^{p-1} = 1 \mod p$.*

Teorema 3.2. *Suponha que p e q sejam primos distintos, $n \in \mathbb{Z} = pq$ e m igual a função de Euler, $m = \varphi(n) = (p-1)(q-1)$. Se a e b são inteiros que satisfazem $ab = 1 \mod m$, então $x^{ab} = x \mod n$ para todo $x \in \mathbb{Z}$.*

3.1 *Rivest, Shamir, Adleman (RSA)*

Este algoritmo é baseado no problema de fatoração por inteiros de produtos de dois números primos grandes [6]. O RSA é o mais famoso sistema criptográfico assimétrico de chave pública. Como este algoritmo é relativamente lento, é indicado que seus usuários, ao invés de usarem diretamente o RSA para criptografar seus dados, usem-no apenas para transmitir chaves criptográficas cifradas que serão empregadas em algoritmos simétricos. Esses algoritmos simétricos permitem que os processos de cifrar e decifrar ocorram de maneira mais rápida. Abaixo descrevemos o algoritmo de RSA.

Geração das Chaves: sejam p e q números primos gerados aleatoriamente, consideramos $n = pq$ e $\varphi(n) = (p-1)(q-1)$. Seja e tal que $\text{mdc}(e, \varphi(n)) = 1$. Pelo algoritmo de Euclides temos que $d = e^{-1} \mod \varphi(n)$. Assim, obtemos as respectivas chaves deste esquema:

A chave pública: (e, n).

A chave secreta: (d, n).

Cifrar: considere m a mensagem a ser cifrada e M o conjunto de todas as mensagens a serem cifradas. Utilizaremos o par de chave pública (e, n) para cifrar m

$$E(m) = m^e \mod n, \ \forall \ m \in M. \tag{1}$$

DOI: 10.5540/03.2022.009.01.0306

Decifrar: a mensagem m pode ser recuperada usando o par de chave secreta (d, n) da seguinte maneira

$$D(E(m)) = E(m)^d \mod n = m. \tag{2}$$

Homomorfismo: sejam m_1 e m_2 mensagens então,

$$\begin{aligned} E(m_1) \cdot E(m_2) &= (m_1^e \mod n) \cdot (m_2^e \mod n) \\ &= (m_1 \cdot m_2)^e \mod n \\ &= E(m_1 \cdot m_2). \end{aligned} \tag{3}$$

Percebemos que este método não permite a adição Homomórfica em mensagens cifradas, ou seja, o RSA é apenas Homomórfico sobre a multiplicação.

Exemplo 3.1. *Vamos considerar que Alice deseja enviar uma mensagem para Bob, e que esta mensagem seja m =CNMAC, assim necessitamos gerar as chaves para cifrarmos e decifrarmos a mensagem corretamente.*

Para a Geração das chaves:

1) escolhemos de forma aleatória dois números primos grandes, $p = 11$ *e* $q = 23$*;*

2) calculamos $n = p.q = 11.23 = 253$*, e a função de Euler* $\varphi(n) = (p-1)(q-1) = (11-1)(23-1) = 220$*;*

3) escolhemos um inteiro e tal que, $1 < e < \varphi(n)$*, de maneira que e e* $\varphi(n)$ *sejam relativamente primos entre si. Por exemplo,* $e = 7$*, pois* $mdc(e, \varphi(n)) = (7, 220) = 1$*;*

4) utilize o algoritmo de Euclides para o cálculo $d = e^{-1} \mod \varphi(n)$*, ou seja, d é o inverso multiplicativo de e* $\mod \varphi(n)$*, ou seja,* $7 \mod 220$*, ver tabela 1.*

Tabela 1: Tabela para o cálculo do Algoritmo de Euclides.

Linha	Q	R	U	V
-1	-	220	1	0
0	-	7	0	1
1	31	3	1	-31
2	2	1	-2	63

Logo, $220(r) + 7(d) = 1 \longrightarrow 220(-2) + 7(63) = 1 \longrightarrow d = 63$.

Logo, a chave pública é dada por $(n, e) \longrightarrow (253, 7)$*, e a chave privada é o par* $(63, 253)$.

Além disso, iremos utilizar a seguinte bijeção $\sigma : L \longrightarrow \mathbb{Z}_{26}$*, onde,* $L = A, B, C, \cdot, X, Y, Z$*. O símbolo* σ *representa as letras do alfabeto e é definido como* $\sigma(A) = 0, \sigma(B) = 1, \ldots, \sigma(Z) = 25$*, para transformar as letras em números inteiros. As correspondências para cada valor* σ *estão listadas na tabela abaixo.*

Figura 2: Tabela para correspondência de mensagem.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Com base na tabela 2, iremos codificar a nossa mensagem em números, assim

DOI: 10.5540/03.2022.009.01.0306

Tabela 2: Tabela da troca de mensagem.

C	N	M	A	C
2	13	12	0	2

Para cifrarmos a mensagem, onde $n-1$, usamos a chave pública da Alice, e executamos uma potenciação modular, assim,

$$E(m) \equiv (CNMAC)^7 \mod 253.$$

A mensagem então pode ser transmitida por um canal inseguro para o Bob,

$$\begin{cases} 2^7 = 128 \mod 253, \\ 13^7 = 216 \mod 253, \\ 12^7 = 177 \mod 253, \\ 0^7 = 0 \mod 253, \\ 2^7 = 128 \mod 253. \end{cases} \tag{4}$$

A mensagem cifrada é $\{128, 216, 177, 0, 128\}$. Se utilizarmos a tabela 2, teremos a seguinte mensagem $E(m) = YIVAY$.

Para decifrarmos a mensagem usando a chave privada, basta fazer outra potenciação modular,

$$E(m)^{63} \equiv (CNMAC) \mod 253.$$

Assim,

$$\begin{cases} 128^{63} = 2 \mod 253, \\ 216^{63} = 13 \mod 253, \\ 177^{63} = 12 \mod 253, \\ 0^{63} = 0 \mod 253, \\ 128^{63} = 2 \mod 253. \end{cases} \tag{5}$$

A mensagem decifrada é $\{2, 13, 12, 0, 2\}$, utilizando a tabela 2, temos que $m =$CNMAC.

Para aplicarmos o Homomorfismo vamos considerar as menagens $m_1 = CNMAC$ e $m_2 = 2022 = CACC$ (substituímos os números pelo alfabeto, ver tabela 2) então pelo algoritmo de RSA,

$$\begin{aligned} E(m_1) \cdot E(m_2) &= ((CNMAC)^7 \mod 253)) \cdot ((CACC)^7 \mod 253) \\ &= ((CNMAC) \cdot (CACC))^7 \mod 253 \\ &= E((CNMAC) \cdot (CACC)). \end{aligned} \tag{6}$$

Se o algoritmo de RSA permitisse o Homomorfismo aditivo, então

$$\begin{aligned} E(m_1) \cdot E(m_2) &= ((CNMAC)^7 \mod 253)) + ((CACC)^7 \mod 253) \\ &= ((CNMAC) + (CACC))^7 \mod 253 \\ &= E((CNMAC) + (CACC)). \end{aligned} \tag{7}$$

4 Considerações Finais

Em álgebra abstrata, o conceito de homomorfismo é muito utilizado quando consideramos estruturas algébricas complicadas isomorfas a estruturas mais simples. Nesse trabalho apresentamos

DOI: 10.5540/03.2022.009.01.0306

de forma resumida uma utilidade mais prática de homomorfismo. A criptografia homomórfica é uma técnica importante pois, além de preservar as operações (aditivas e multiplicativas), decifra corretamente as mensagens ou os dados utilizados. Deste modo, com o auxílio do homomorfismo, é possibilitada a comunicação segura entre os pares envolvidos.

Agradecimentos

Agradeço a agência de fomento CNPQ e a Petrobras pelos auxílios financeiros.

Referências

[1] Adilson Gonçalves. **Introdução à Álgebra**. 3nd. Projeto Euclides- IMPA, 1995.

[2] Shai Halevi. "Homomorphic Encryption". Em: **Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich**. Cham: Springer International Publishing, 2017, pp. 219–276. ISBN: 978-3-319-57048-8. DOI: 10.1007/978-3-319-57048-8_5. URL: https://doi.org/10.1007/978-3-319-57048-8_5.

[3] R. Klima, N. Sigmon e Stitzinger E. "Applications of Abstract Algebra with MAPLE". Em: 1nd. CRC Press, 1999.

[4] Adriana López-Alt, Eran Tromer e Vinod Vaikuntanathan. "On-the-fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption". Em: **Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing**. STOC '12. New York, New York, USA: ACM, 2012, pp. 1219–1234. ISBN: 978-1-4503-1245-5.

[5] Pascal Paillier. "Trapdooring Discrete Logarithms on Elliptic Curves over Rings". Em: **Advances in Cryptology — ASIACRYPT 2000**. Ed. por Tatsuaki Okamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 573–584. ISBN: 978-3-540-44448-0.

[6] R L Rivest, L Adleman e M L Dertouzos. "On Data Banks and Privacy Homomorphisms". Em: **Foundations of Secure Computation, Academia Press** 4 (1978), pp. 169–180.

[7] Sattam S. Al-Riyami e Kenneth G. Paterson. "Certificateless Public Key Cryptography". Em: **Advances in Cryptology - ASIACRYPT 2003**. Ed. por Chi-Sung Laih. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 452–473. ISBN: 978-3-540-40061-5.

[8] Alok Sinha. "Client-Server Computing". Em: **Commun. ACM** 35.7 (jul. de 1992), pp. 77–98. ISSN: 0001-0782. DOI: 10.1145/129902.129908. URL: https://doi.org/10.1145/129902.129908.

